

Challenges in Distance-Bounding

Ioana Boureanu¹ and Serge Vaudenay²

¹ Akamai Technologies Limited
EMEA HQ, UK
<http://people.itcarlson.com/ioana>
² EPFL
Lausanne, Switzerland
<http://lasec.epfl.ch>

Abstract. Smartcard-based protocols represent an increasingly large share of the wireless authentication market, from contactless payments to remote car unlocking. Unfortunately, relay attacks pose a significant threat to these wireless solutions. However, this risk can be mitigated through the use of distance-bounding protocols. In this paper, we discuss the core challenges for distance-bounding, in terms of both theoretical and practical considerations. We focus on their security, but we also explore the difficulties encountered in their design and implementation. Moreover, we present our vision of the future of these protocols and of the possible paths towards their secure deployment.

Keywords: distance-bounding, provable security

In the well-known chess grandmaster problem, there are two grandmasters A and B , and a mere player C , who in fact does not know so much about chess. Grandmaster A is playing “white” against C , and grandmaster B is playing “black” against C . C employs the following strategy. Whenever A makes a move, C goes to the other chessboard—where he is playing against B —and replicates A ’s latest move. Then, C waits for B ’s move and reproduces it on the board where he is playing against A , and so on. Consequently, both A and B will have a hard time playing against C , even if C is no chess expert. In this way, if there is no draw, C will win in front of one grandmaster. And, what C did in his devious strategy is nothing else but a typical *relay attack*.

Nowadays, we have RFID keys to open our cars in a contactless way and, even start their engines in the same remote manner. We carry NFC credit cards in our (real and virtual) wallets and, more and more often, we use them to make contactless payments. All these use protocols between an RFID device, which is often called a *prover* (i.e., the card), and a reader, which is often referred to as a *verifier*. Typically, the aforementioned protocols require no real input, going from the (human) carrier towards this whole RFID system (i.e., we just hold a card close enough to the reader, without any further prompts for approval or credentials). So, these protocols are prone to relay attacks. I.e., the signals of an honest prover/card would be captured by a malicious reader, which in turn will send these signals to a malicious prover. The latter will replicate this further to a honest verifier/reader. In this way, an attacker can, e.g., pay with someone else’s resources on a valid terminal, without the victim’s or the terminal’s suspicions.

To defeat this attack, one solution consists in verifying the *proximity* between the authenticating prover and the verifier. This would assume that a correct execution of the protocol is done from precise locations, or—easier—that it is all run within a bounded area. I.e., if an intended card is close enough to a reader, then the communication between the two should happen fast enough (and this is measurable by the verifier). Indeed, if the interactions are to happen rapidly enough, then relaying by adversarial parties should become virtually impossible.

Proving proximity can be achieved through *distance-bounding (DB)* protocols, which were introduced by Brands and Chaum in 1993 [8]. The original idea came from Beth and Desmedt (1990) and

it was that of enforcing certain security guarantees by measuring the communication-time. Within one DB protocol run, the prover demonstrates that he is in the proximity of a verifier. In DB, the prover also authenticates himself to the verifier.

Distance-bounding is in itself prone to some specific attacks. In a *distance-fraud (DF)*, a prover tries to convince the verifier that he is closer than he really is. In a *mafia-fraud attack*, an adversary tries to demonstrate to an honest verifier that an honest prover is in the verifier's proximity although the prover is in reality far away from the verifier. The adversary could consist of two communicating bodies located in the proximity of the prover and the verifier. This threat generalizes relay attacks, in which the adversary passively relays messages. For instance, the prover is buying something in a shop in which the terminal has been tampered with (by the "mafia"). This terminal is relaying the communication to a malicious card carrying out a more expensive transaction in a fully legitimate shop (with a terminal denoting an honest verifier). This type of attack is also often called the *man-in-the-middle (MiM)* attack. In a *terrorist-fraud (TF)*, the adversary has the same goal as in the mafia-fraud attack, but in this case the prover is dishonest and colludes with the adversary. There are even extensions of these attacks such as *distance hijacking*. This fraud involves one dishonest, far-away prover and several honest provers, without the latter colluding with the former.³

Driven by the need to defeat relay attacks, distance bounding received more attention recently. It appears very probable that it will be integrated in existing infrastructures. However, many challenges lie ahead. In the following, we report the main such difficulties in the area of DB.

Challenge 1: Implementing the Time-Critical DB Phase

This challenge is intrinsic to wireless media and relaying (thus to implementation of DB, as well). It can be summarized as follows. The computation delays encountered by, e.g., RFID tags, are orders of magnitudes higher than the communication speeds between such a tag and a reader. This difference leaves room for possible relaying, which would be hard to detect if we were to measure/bound the communication times as we do in DB.

Concretely, a DB protocol starts with an initialization phase. Then, it goes through a succession of n rounds. In each round, the verifier sends a challenge and waits for a response to arrive back within a time bound. This waiting time corresponds to the round trip time of flight to the maximal allowed distance to the prover, plus a small overhead due to latency. This so-called *distance-bounding* phase is time-critical and it obviously imposes very fast computations at each end, typically of less than a single clock cycle per round. (Every bit must be treated on the fly, upon arrival, with no delay, and there is virtually no time for extra computations.) But is this achievable in practice? Can, e.g., RFID-based DB protocols have low enough computation latencies?

Let us then estimate some of these bounds, time-overheads, latencies, etc. Firstly, we know that the verifier will accept a honest prover running the protocol from a distance up to some bound; let us denote this bound as B_{close} . Therefore, the (idealized) round trip time of flight between the prover and the verifier is up to $\frac{2}{c}B_{\text{close}}$, where c is the speed of light. A honest prover like above, assuming it is implemented using a common RFID tag, will certainly need some time to process the reception of a challenge, to compute the response, and to process the sending of it; let us denote the total of this time as t_{overhead} . So, from the verifier viewpoint, the whole process (from the release of the challenge to the arrival of the response) must take

$$\frac{2}{c}B_{\text{close}} + t_{\text{overhead}}.$$

³ We refer the interested reader to [7] for more details.

Secondly, it is the case that the verifier will reject a malicious prover running the protocol from a point that is beyond a bound, which we denote as B_{far} . There may be a difference $\Delta t_{\text{overhead}}$ in the overhead time t_{overhead} of the honest prover and the one of the malicious prover. So, the round-trip time will then be of

$$\frac{2}{c}B_{\text{far}} + t_{\text{overhead}} - \Delta t_{\text{overhead}}.$$

Because the malicious prover could use some expensive equipment to make this time much smaller, we consider the worst case

$$\Delta t_{\text{overhead}} \approx t_{\text{overhead}}.$$

Hence, to make the protocol feasible, we must have

$$t_{\text{overhead}} \leq \frac{2}{c}(B_{\text{far}} - B_{\text{close}}).$$

Now, let us take an example: consider the case of wireless payment at the cashier of a supermarket. A customer queuing at the cashier, 1m away from the payment terminal, may be the victim of a relay attack. Then, to protect him, we shall have $B_{\text{far}} \leq 1\text{m}$. When he wants to pay, with his contactless credit card touching the payment terminal, the protocol should nonetheless work with $B_{\text{close}} \geq 1\text{cm}$. So, we must have $t_{\text{overhead}} \leq 6.6\text{ns}$. This time is very short. To have an idea of how short this is, just think of the fastest transmission rate of WiFi routers reaching 600 Mbps. After sending one bit, the waiting time to send a second bit is as long as 1667ns. So, we do not even have time to send bits in sequence. Likewise, this very short t_{overhead} does not allow a lot of computation on the prover's side. Typically, the response is precomputed in a table. This imposes that the challenge is *very* short: typically, 1 or 2 bits. The usual transmission rate also imposes that all bits of the challenge are sent at the same time. As we said above, the typical “duration of a bit” is measured in microseconds, which is way too long. Finally, the reception device must interpret the bit as soon as it “starts to arrive” and not wait until the entire duration of the bit. Actually, as a parallel, the analog-digital conversion is typically too slow and we may have no time to do it. So, we would rather do the “computation” of the response in analogue mode directly.

This has another consequence: interpreting the challenge and sending the response is very likely to make errors. So, a challenge/response round may contain a significant amount of communication noise.

All of these time constraints on communication vs. computation seem to raise a real challenge to any real-life DB implementation.

Challenge 2: Identifying the Security Threats

As we mentioned above, three main types of DB attacks have been distinguished: *distance-fraud*, *mafia-fraud* and *terrorist-fraud*, following a terminology due to Desmedt. (See the description of these frauds on p. 2.)

Like with other threat-models, some of these attack-scenarios for DB are less realistic than others (e.g., mafia-fraud is the most credible, to our opinion). But, each could pose a real threat in some (specific) environment. At the same time, as we will explain below, the DB community witnesses that protecting against all these threats (at once) is cumbersome. So, the question associated to this challenge is: “Whilst some frauds are harder to mount & overall protection seems hard/costly, could real-life security live without protection against certain DB frauds?”. Here are some discussions towards this question...

Terrorist-fraud...? First of all, we could wonder why should one even worry about terrorist-frauds (TF) at all. Recall that in such fraud, the prover P is malicious and far away, and colludes with a close-by adversary A to make the verifier V think that P is close to him. To fool V like this, there is a trivial way: the prover P simply gives his secret-key to the adversary, so that A would run the protocol on behalf of P . People think that there is no need to protect against it because malicious provers would certainly not be willing to share their secret. So, this attack is not considered as a valid terrorist-fraud. To be terrorist-fraud-resistant, the DB protocol-designs should make sure that helping A pass the protocol entails A deducing P 's secret key. So, if such a helped attack occurs, then the key-leakage reduces it to trivial one above, which – as we said– is considered an *invalid* terrorist-fraud, so the protocol is not TF-vulnerable. Concretely, a terrorist-fraud is valid if A passes the protocol but does *not* deduce P 's secret.

The most common technique to protect against terrorist frauds consists in making the i th bit of the secret leak from the prover's response-function table, in the i round. Indeed, in each round of the DB phase, the prover calculates a response from the received challenge. Having A pass the protocol with a good probability implies that A must know the response to all possible challenges in each round. I.e., A must know the full table of the response functions. Typically, the XOR of all entries in the table of the i th round is the i th bit of the secret, making A be able to deduce it.

As an example of such attack, imagine that P runs the time-insensitive phases by himself and gives, for each of the n time-critical rounds, a table of the response-function for A to use, so that A can quickly reply to V , for any of V 's challenges. We consider an attack in which P gives the correct table of the response-function in the first $(1 - \theta)n$ rounds. For the next θn rounds, he corrupts one random (challenge) entry in the table by replacing it with a random response (which may be correct or not). Clearly, A passes the protocol if the verifier does not ask for any of these challenges, or if he does but the random, altered response is correct. So, A can infer the secret in which θn bits are randomly corrupted. If θ is such that recovering θn bits by exhaustive search is intractable, then such attack does not leak the secret and is considered a valid terrorist fraud.

There are many (good) protocols which do not offer any form of resistance to this threat. Furthermore, some authors, e.g., Hermans *et al.* [14], even argue that resisting terrorist-fraud somehow weakens security. Indeed, designing the protocol such that it leaks the secret in the case of such an attack is a dangerous approach. However, we believe this weakening does not need to be the case. We think that security proofs are there to assure that such designs (or leaks) pose no risk to honest provers. To some extent, the lack of practical terrorist frauds could dismiss the interest in TF-protection. But, we believe that one should rather ensure security against this type of exotic attack and thus cover for practical scenarios that researchers have not yet imagined; such protection should of course come at a reasonable price.

Distance-fraud...? We may also wonder about the need for distance-fraud resistance: i.e., should we really worry about malicious provers at all? In an ideal world where all provers are honest, we could simply focus on protection against man-in-the-middle attacks, which encompass impersonation, relaying, etc. These latter attacks were the original motivation and remain the most important question to address. Still, we think that it makes sense to imagine that malicious provers could exist and could try to abuse a system. For instance, in payment systems, liability is an important question. In such a setting, we could imagine a scenario where a prover would attack the system, then deny having made a payment; he could use the guarantees of distance bounding (which he would have managed to falsify) to say that he was too far to have made that payment. On a similar note, in contactless payment system, if a prover manages to illicitly pay from far away, he could turn the previous argument around and then use this further as an alibi to pretend he was near to the verifying point-of-sale. Conversely,

someone with a stolen payment token could run a distance fraud to pay from far away. So, security against malicious provers shall not be neglected.

Challenge 3: Building up a Secure Protocol

In general, attempts to construct all-fraud-resistant distance-bounding protocols have proven flawed. Some authors even suggested it would be impossible [1]. In Table 1, page 11, the popular distance-bounding protocols and their vulnerabilities as best-known up to 2013 are reported. In this table, n is the number of DB rounds and θ is a parameter of a terrorist-fraud (TF) such that recovering θn bits by exhaustive search is intractable. As we can see, all protocols but two are vulnerable to at least one of the classical threats (i.e., one probability is equal to 1), or some instances of the protocol are (as explained below).

Sometimes, DB protocols are based on an underlying primitive called a *pseudorandom function* (PRF). The PRF is unspecified and security is assumed to depend on the PRF assumption. Some authors heavily use arguments such as “if f is a pseudorandom function (PRF), then this protocol is secure against...”. In fact, in a line by Boureanu *et al.* [3], it was proven that, if PRFs exist, then such statements used in semi-formal proofs are incorrect. When employed with some specific (artificially backdoor-ed) PRFs, many protocols were shown to be indeed vulnerable to distance-fraud and/or man-in-the-middle attacks. This result appears in Table 1 in the entries “ p to 1”, as the success of the attack varies from very low to 1, with the choice of the PRF. Except for these artificial PRFs, these attacks may become impractical, although there is no formal proof of such statement.

In a parallel line, Kim *et al.* [15] proved in 2008 that many existing distance-bounding protocols are also subject to mafia-fraud. In doing so, they exposed for the first time the importance of the return channel in the (in)security of these protocols; the presence or the lack of a return channel respectively model the simple facts whether the adversary is or is not able to observe the output of the protocol (e.g., whether we see or we do not see a LED turning green or red on a card reader, suggesting the success or the failure of an access protocol).

Finally, Hancke [12] raised the alarm on the fact that noisy communications and tolerance to them must also be addressed in the security analysis. Indeed, Hancke showed that almost all protocols in Table 1 (all except the SKI and Fischlin-Onete protocols) are vulnerable to terrorist-frauds when they tolerate noisy conditions.

The figure shows a dire situation, so the question of *provable security* against all frauds mounted has been standing prominently in the last two years. In the last year however, two (classes of) provably secure DB protocols, the SKI class (2013) [5] and the Fischlin-Onete protocol (2013) [11] have been published.

Challenge 4: A Formal Security Model

Several models for DB security have been published recently. In [1], Avoine *et al.* first give a complete, but rather informal model for distance-bounding in 2011. They define distance-bounding as the combination of authentication and distance-checking.

A more mature model for distance-bounding was presented by Dürholz *et al.* in [9] in 2011. It presents a communication model in which notions of time or distance are only implicit. It requires specifying protocols by explicitly distinguishing a lazy phase and a time-critical one. This model formalizes the three classical types of frauds and an extra notion of *impersonation fraud*. The threat models are very specific, presented in terms of protocol session interleaving. That is, each type of

fraud must exhaustively specify what kinds of interleavings are not allowed (what is called *tainted session* in [9]), leading us to many variants. Possibly due to this specificity and to their security requirements being possibly too strict, the model is too strong, admitted by its authors in 2013 [10]. In this model, certain insecurities (impersonation or terrorist-fraud) are hard-to-defend claims, leading to unconvincing attacks. Fischlin and Onete later designed [11] a DB protocol, proven secure in their model from [9].

In 2013, Boureanu *et al.* [4] introduced a formal model for distance-bounding, based on techniques and definitions similar to those of interactive proofs. This model seems natural, since distance-bounding can indeed be viewed like an interactive proof (of proximity). Along with the formal model, Boureanu *et al.* introduced, in 2013 [5,4], the class of provably secure DB protocols called SKI.

To sum up (especially looking at challenge 4 and challenge 5), we can see that in the 20-year old DB literature informal DB discussions and insecure protocols prevail, and that the formal models for DB security has just started to take shape.

Challenge 5: Efficiency

Another question, which has not been risen before, is that of the performances of these two surviving, provably secure protocols: SKI and Fischlin-Onete. By this, we mean the following: what is the price to pay, in terms of number n of rounds of the DB phase, in order to achieve different security guarantees? Are the protections of different frauds really conflicting in terms of communication/computation costs? Is, e.g., TF-resistance coming at a too high price in terms of protocol rounds? We looked at answering such questions, and we summarize our findings in Figure 1, on page 11. This figure is to be read as follows. On the x -axis we have the number of DB rounds and on the y -axis the level of security associated. I.e., if we look at the blue line, representing MiM attacks on SKI, at its represented peak we read that an “a MiM attack on SKI that would defeat 40-bit security would require $n = 150$ rounds”.

To make this plot, we arbitrarily assumed that challenge/response rounds fail due to noise with probability $p_{\text{noise}} = 5\%$, and we tuned the minimal number τ of correct rounds so that the false rejection rate (FRR) would be of $p_{\text{FRR}} = 1\%$. The values used/obtained are linked by

$$p_{\text{FRR}} = \text{Tail}(n, \tau, 1 - p_{\text{noise}}),$$

using the tail of the binomial distribution

$$\text{Tail}(n, \tau, \rho) = \sum_{i=\tau}^n \binom{n}{i} \rho^i (1 - \rho)^{n-i}.$$

The upper bound of the probability for a MiM attack onto SKI is

$$p_{\text{MiM}} = \text{Tail}\left(n, \tau, \frac{2}{3}\right).$$

We recall that

$$\lim_{n \rightarrow +\infty} -\frac{1}{n} \log_2 \text{Tail}(n, tn, \rho) = \frac{(t - \rho)^2}{2\rho(1 - \rho)}.$$

So, the slope of the line approaching the SKI MiM curve is

$$\eta_{\text{MiM}} \approx 0.18.$$

Increasing the number of rounds by Δn would result in increasing security against MiM by $\eta_{\text{MiM}} \cdot \Delta n$ bits.

Similarly, the slopes of the SKI DF curve (which is also the DF and MiM curve for the Fischlin-Onete protocol) and the SKI TF curve are

$$\eta_{\text{DF}} \approx 0.11, \quad \eta_{\text{TF}} \approx 0.06.$$

As the graph in Figure 1, page 11, shows SKI requires 151 rounds for a DF-resistance of 2^{-20} (i.e., the probability to have a successful DF over 151 DB rounds of SKI is 2^{-20}), 91 rounds for a MiM-resistance of 2^{-20} , and 315 rounds for a TF-resistance of 2^{-20} .

Both for the SKI and the Fischlin-Onete protocols, the required number of rounds is still large.

But what would it mean for a DB protocol to offer optimal security? Note that a DB protocol expecting τ correct rounds out of n binary-challenge rounds is always vulnerable to the following MiM attack: the man-in-the-middle guesses c_i before it arrives and asks for r_i from the honest prover. When c_i arrives from the verifier, he can answer if his guess is correct. Otherwise, he sends a noisy-like answer and continues the protocol. This attack works with probability

$$p_{\text{MiM}} = \text{Tail}\left(n, \tau, \frac{1}{2}\right).$$

This corresponds to a MiM slope of

$$\eta \approx 0.41.$$

Since this attack will always exist, the curve above is the optimal curve in terms of MiM-protection, for protocols with binary challenges. This is actually reached by the recent DB2 protocol from [7]. So, we could drastically reduce the number of rounds of the SKI and the Fischlin-Onete protocols. By comparison, DB2 needs $n = 43$ rounds for a MiM-security of 2^{-20} . But the $n = 123$ rounds that it needs for DF-security is suboptimal. So, the race for the optimal protocol is going on. Even though some protocols are optimal with respect to one security metric, they are not for all of them and the correct trade-off must be identified.

Challenge 6: Public-Key Distance Bounding

Distance-bounding protocols in the literature are based on a shared key: they assume that the prover and the verifier share a secret. In many applications, such an assumption is unreasonable. For instance, the wireless payment terminal at the supermarket is unlikely to share a secret with the customer's credit card. It would make more sense that the credit card holds a secret and has some (certified) public key. The payment terminal would only need this public key to verify the proximity proof. Hence, there is a need for public-key based protocols.

In the literature, only three public-key distance bounding protocols exist. There is the original Brands-Chaum protocol (1993) [8]. It does not offer any security against terrorist-fraud. Moreover, the Bussard-Bagga protocol (2004) was completely broken by Bay *et al.* in 2012 [2]. Finally, the Hermans-Onete-Peters protocol (2013) [14] has a security proof but it does not protect against terrorist-fraud. However, it features privacy protection: the verifier has also a public/secret key pair and an outside observer cannot figure out the identity of the prover.

We could however wonder why we have only two public-key distance bounding protocols, all without terrorist-fraud protection. Actually, it is easy to transform any symmetric DB into a public-key one: we just run a key agreement (KA) protocol (such as the famous Diffie-Hellman protocol in

semi-authenticated mode) to set up a symmetric key sym . Then, we run a symmetric DB protocol with key sym . (See Figure 2, on page 11.) The symmetric DB protocol consists of two algorithms $P_{\text{symmetric}}(\text{sym})$ and $V_{\text{symmetric}}(\text{sym})$. This could, for instance, be the Hancke-Kuhn protocol [13]. The KA is semi-authenticated: the prover uses $\text{KA}_P(\text{sk})$, where sk is his long-term secret key, and the verifier uses $\text{KA}_V(\text{pk})$, where pk is the long-term public key of the prover. But, this solution is vulnerable to the terrorist fraud in which the prover just sends sym to the close-by adversary who runs the symmetric DB protocol without having access to sk . Hence, building classical TF-protection into public-key DB does not work.

So, making a provably secure public-key distance-bounding protocol resisting the three types of fraud is still an open problem. Whenever this challenge is solved, making an *efficient* public-key DB protocol will be the next hurdle.

Challenge 7: Integration

Proximity proofs must be integrated in applications, like payment systems, access-control, or remote unlocking scenarios. For this, the development of the appropriate infrastructures is still an open problem. Let us discuss each of the above cases individually.

Secure remote unlocking. The task seems to be easy in the case of remote unlocking of cars using a wireless key. This is because the prover and the verifier therein are stable points in the infrastructure, i.e., there is one verifier/reader on the car that will accept one key belonging to the owner of the car. Then, the only relevant question therein lies in the key distribution problem. But key distribution should/could be completely independent from the proof of proximity protocol (that may use the established key).

Secure access-control. For access-controls settings, e.g., entering/exiting buildings, the task of incorporating DB therein may seem slightly harder. That is because there are several verifiers that may authenticate (at different points) one given prover; i.e., one person's badge will be read by different readers, one on each door that he/she passes through. Clearly, such readers will not store the secret of all provers. However, we can assume that reader on the doors have secure *online* access to an authentication server that holds all the keys.

For such application, implementation could be done in a straightforward manner: the reader (on the door) could just relay communications between the prover and the authentication server, and—as customary—measure the time taken by the challenge/response rounds to forward the timer values to the server. We could however save a few communication steps between the server and the door. For instance, in Figure 3, on page 12, we describe a protocol based on the SKI protocol and requiring only one query/response steps to and from the server. Essentially, the server selects the challenges for the verifier and sends a vector t of the commitments of all possible responses. That is, t_i is a commitment to the response r_i which is expected in the round i . The commitment is to be opened with a decommit key

$$\rho_i = \text{PRF}_x(N_P, N_V, L, i, c_i),$$

(where the parameters are defined in SKI [5,4]) which must be revealed by the prover. This comes however at the cost: the server must compute all commitments and the prover must open them.

Secure contactless payments. In the contactless payment infrastructure, the prover (an NFC creditcard or smartphone) wants to pay the verifier. Verifiers/readers are more pervasive, and we cannot always assume that they have a secure online access to an authentication server. It is not reasonable to assume

that any of these verifiers would have access to the secret of the credit-card either. So, to integrate DB within, we need a solution based on public keys. For instance, we can assume a certificate-based solution. The prover/tag sends the verifier/reader a certificate on his public key. I.e., this is a signature (with extraction) on the prover's public key made by a certification authority (e.g., the bank/bank-group emitting the card), and the verifiers/readers hold the public keys of all (or at least “root”) certification authorities ck (e.g., of all reputable banking groups). Hence, using the latter, the readers can extract the public key of the prover. From here on, the prover and the verifier will run some public-key-based distance bounding. We use a public-key distance bounding protocol DB such as the one by Brands and Chaum, by Hermans, Onete, and Peeters, or one based on Figure 2, on page 11. Such a DB protocol integrates in a contactless payment scheme with a certificate infrastructure: a credit-card provides a certificate from the bank to the seller. The seller checks it and extracts the public key of the credit-card. Then, they just run the distance-bounding protocol to authenticate the credit-card and validate its proximity.

Conclusions

In this paper, we discussed the core challenges in distance-bounding, in terms of both theoretical (e.g., security models) and practical considerations (e.g., implementation difficulties). We also presented our vision for the future of these protocols, the possible and advisable paths towards their secure deployment. We focused on the security of DB, but we also explored the problems encountered in their design, implementation and in acquiring efficiency; overall, we underlined that these four aspects do not harmoniously coexist. As a summary, the focus was on the current co-dependent challenges to face today if one wanted to design DB protocol that is: 1). provably secure; 2) efficient; 3). implementable in practical wireless media. These three points above are relevant to researchers and practitioners alike.

References

1. G. Avoine, M. Bingöl, S. Kardas, C. Lauradoux, B. Martin. A Framework for Analyzing RFID Distance Bounding Protocols. *Journal of Computer Security*, vol. 19(2), pp. 289–317, 2011.
2. A. Bay, I. Boureanu, A. Mitrokotsa, I. Spulber, S. Vaudenay. The Bussard-Bagga and Other Distance-Bounding Protocols under Attacks. In *Proceedings of the 8th International Conference of Information Security and Cryptology (IN-SCRYPT'12)*, Beijing, China, Lecture Notes in Computer Science 7763, pp. 371–391, Springer-Verlag, 2012.
3. I. Boureanu, A. Mitrokotsa, S. Vaudenay. On the Pseudorandom Function Assumption in (Secure) Distance-Bounding Protocols - PRF-ness alone Does Not Stop the Frauds! In *Proceedings of the 2nd International Conference on Cryptology and Information Security in Latin America (LATINCRYPT'12)*, Santiago, Chile, Lecture Notes in Computer Science 7533, pp. 100–120, Springer-Verlag, 2012.
4. I. Boureanu, A. Mitrokotsa, S. Vaudenay. Practical & Provably Secure Distance-Bounding. Proceedings of the 16th Information Security Conference (ISC 2013), Dallas, Texas, USA, to appear
5. I. Boureanu, A. Mitrokotsa, S. Vaudenay. Secure & Lightweight Distance-Bounding. In *Proceedings of the Second International Workshop in Lightweight Cryptography for Security and Privacy (LIGHTSEC'13)*, Gebze, Turkey, Lecture Notes in Computer Science 8162, pp. 97–113, Springer-Verlag, 2013.
6. I. Boureanu, A. Mitrokotsa, S. Vaudenay. Towards Secure Distance Bounding. Proceedings of the 20th International Workshop on Fast Software Encryption (FSE' 13), Singapore, 55–67, 2013
7. I. Boureanu, S. Vaudenay. Optimal Proximity Proofs. Eprint technical report, 2014. <http://eprint.iacr.org/2014/693.pdf>
8. S. Brands, D. Chaum. Distance-Bounding Protocols (Extended Abstract). In *Advances in Cryptology EUROCRYPT'93*, Lofthus, Norway, Lecture Notes in Computer Science 765, pp. 344–359, Springer-Verlag, 1994.
9. U. Dürholz, M. Fischlin, M. Kasper, C. Onete. A Formal Approach to Distance-Bounding RFID Protocols. In *Proceedings of the 14th Information Security Conference (ISC'11)*, Xi'an, China, Lecture Notes in Computer Science 7001, pp. 47–62, Springer-Verlag, 2011.

10. M. Fischlin, C. Onete. Subtle Kinks in Distance-Bounding: an Analysis of Prominent Protocols. In *Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WISEC'13)*, Budapest, Hungary, pp. 195–206, ACM, 2013.
11. M. Fischlin, C. Onete. Terrorism in Distance Bounding: Modelling Terrorist-Fraud Resistance. In *Proceedings of the 11th International Conference on Applied Cryptography and Network Security (ACNS'13)*, Banff AB, Canada, Lecture Notes in Computer Science 7954, pp. 414–431, Springer-Verlag, 2013.
12. G.P. Hancke. Distance Bounding for RFID: Effectiveness of Terrorist Fraud. In *Proceedings of the 3rd IEEE International Conference on RFID-Technology and Applications (RFID-TA'12)*, Nice, France, pp. 91–96, IEEE, 2012.
13. G.P. Hancke, M.G. Kuhn. An RFID Distance Bounding Protocol. In *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm'05)*, Athens, Greece, pp. 67–73, IEEE, 2005.
14. J. Hermans, R. Peeters, C. Onete. Efficient, Secure, Private Distance Bounding without Key Updates. In *Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WISEC'13)*, Budapest, Hungary, pp. 195–206, ACM, 2013.
15. C.H. Kim, G. Avoine, F. Koeune, F.-X. Standaert, O. Pereira. The Swiss-Knife RFID Distance Bounding Protocol. In *Proceedings of the 11th International Conference on Information Security and Cryptology (ICISC'08)*, Seoul, Korea, Lecture Notes in Computer Science 5461, pp. 98–115, Springer-Verlag, 2009.

Protocol	Success Probability		
	Distance-Fraud	Man-in-the-middle (Mafia-Fraud)	Collusion-Fraud (Terrorist-Fraud)
Brands & Chaum (1993)	$(1/2)^n$	$(1/2)^n$	1
Bussard & Bagga (2004)	1	$(1/2)^n$	1
Čapkun <i>et al.</i> (2003)	$(1/2)^n$	$(1/2)^n$	1
Hancke & Kuhn (2005)	$(3/4)^n$ to 1	$(3/4)^n$	1
Reid <i>et al.</i> (2007)	$(3/4)^n$ to 1	1	$(3/4)^{\theta n}$
Singelée & Preneel (2007)	$(1/2)^n$	$(1/2)^n$	1
Tu & Piraumuthu (2007)	$(3/4)^n$	1	$(3/4)^{\theta n}$
Munilla & Peinado (2008)	$(3/4)^n$	$(3/5)^n$	1
Swiss-Knife (2008)	$(3/4)^n$	$(1/2)^n$ to 1	$(3/4)^{\theta n}$
Kim & Avoine (2009)	$(7/8)^n$	$(1/2)^n$	1
Nikov & Vaclair (2008)	$1/k$	$(1/2)^n$	1
Avoine <i>et al.</i> (2011)	$(3/4)^n$ to 1	$(2/3)^n$ to 1	$(5/6)^{\theta n}$
SKI (2013)	$(3/4)^n$	$(2/3)^n$	$(5/6)^{\theta n}$
Fischlin & Onete (2013)	$(3/4)^n$	$(3/4)^n$	$(3/4)^{\theta n}$

Table 1. Best-known attacks on prominent distance-bounding protocols (taken from [6, Table 1],[5, Table 1], where we corrected herein the terrorist-fraud entries for Avoine *et al.* and for SKI)

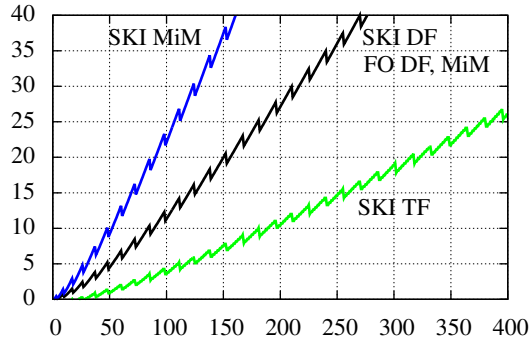


Fig. 1. Security levels (in bitlength-equivalent security) in terms of the number of rounds

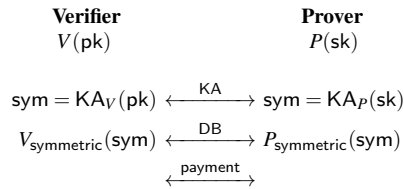


Fig. 2. A public-key DB from a symmetric one

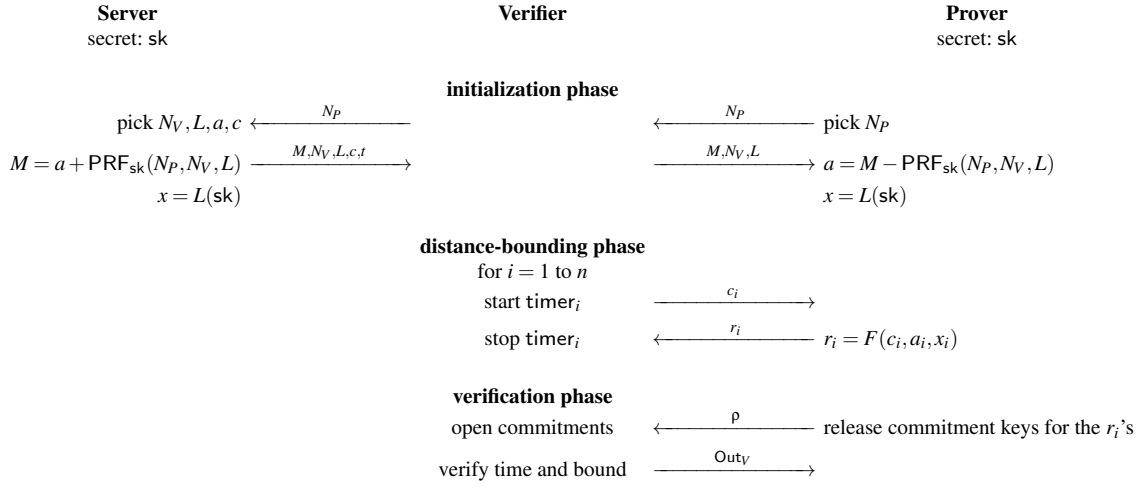


Fig. 3. DB integrated in access-control based on SKI